PCT/DE 03/02274

BUNDESREPUBLIK DEUTSCHLAND

Prioritätsbescheinigung über die Einreichung einer Patentanmeldung

Aktenzeichen: 102 31 971.5

Anmeldetag: 15. Juli 2002

Anmelder/Inhaber: Siemens Aktiengesellschaft, München/DE

Bezeichnung: Verfahren und Vorrichtung zum Encodieren/ Decodieren von strukturierten Dokumenten, insbesondere XML-Dokumenten

IPC: G 06 F 17/21

Die angehefteten Stücke sind eine richtige und genaue Wiedergabe der ursprünglichen Unterlagen dieser Patentanmeldung.

München, den 23. Juli 2003
Deutsches Patent- und Markenamt
Der Präsident
Im Auftrag

Ebert

DE 102 31 971.5 vom 15.07.2002

2002 P 11264

Beschreibung

Verfahren und Vorrichtungen zum Encodieren/Decodieren von strukturierten Dokumenten, insbesondere von XML-Dokumenten

Die Erfindung betrifft Verfahren bzw. Vorrichtungen zum Encodieren von strukturierten Dokumenten, insbesondere XML-Dokumenten, bei denen aus einem strukturierten Dokument in Abhängigkeit eines Schemas ein Bitstrom erzeugt wird und ein Verfahren bzw. eine Vorrichtung zum Decodieren, bei denen aus einem Bitstrom in Abhängigkeit eines Schemas ein strukturiertes Dokument erzeugt wird.

Im Rahmen der Arbeit am MPEG-7 Standard wurde ein Verfahren zur binären Codierung von XML Daten entwickelt, das im folgenden BiM-Verfahren genannt wird und beispielsweise aus der Veröffentlichung ISO/IEC FDIS 15938-1:2001(E), "Information Technology - Multimedia Content Description Interface - Part 1: Systems" bekannt ist. Dieses Verfahren verwendet XML Schema Definitionen, die beim Encoder und Decoder vorliegen, beispielsweise das MPEG-7 Schema, um die Codes für die einzelnen Datenelemente der XML Beschreibung zu generieren. Dieses Verfahren setzt voraus, dass dem Encoder und dem Decoder zumindest teilweise die selben Schemadefinitionen vorliegen. Dies kann beispielsweise gewährleistet werden, indem ein standardisiertes XML Schema im Decoder fest eingebaut wird. Ausserdem besteht die Möglichkeit das Schema separat oder zusätzlich zum eigentlichen Dokument dem Decoder zu übermitteln. Die Übertragung des Schemas vom Encoder zum Decoder kann in textueller Form durchgeführt werden, wobei eine Standard Textkompression, wie z.B. ZIP, angewendet werden kann.

Die der Erfindung zu Grunde liegende Aufgabe besteht nun darin, Verfahren bzw. Vorrichtungen derart anzugeben, dass die Übertragung des Schemas besonders effizient erfolgt und dass die übertragene Datenmenge und die Rechenleistung am

Decoder, die für die Erzeugung der Codetabellen aus dem Schema nötig ist, reduziert wird. Außerdem soll die Konsistenz eines nicht vollständig übertragenen Schemas sichergestellt werden.

Diese Aufgabe wird hinsichtlich des Encodierverfahrens durch die Merkmale des Patentanspruchs 1, hinsichtlich des Decodierverfahrens durch die Merkmale des Patentanspruchs 6 , hinsichtlich der Encodiervorrichtung durch die Merkmale des Patentanspruchs 12 und hinsichtlich der Decodiervorrichtung durch die Merkmale des Patentanspruchs 13 erfindungsgemäß gelöst.

Die weiteren Ansprüche betreffen vorteilhafte Ausgestaltungen der erfindungsgemäßen Verfahren bzw. Vorrichtungen.

Die Erfindung besteht im wesentlichen darin, mit einem Encodierverfahren aus einem Schema in Abhängigkeit eines Metaschemas einen Bitstrom oder einen Teil eines Bitstromes zu erzeugen, wobei eine oder mehrere der folgenden Optimierungen durchgeführt werden:
- Abspaltung von anonymous Types aus Elementdeklarationen und Attributdeklarationen, und Codierung als eigener Typ, dessen Typdefinition als Top-Level Element in der Schema Definition instantiiert ist,
- Normalisierung der Syntax Trees auf Encoderseite,
- Ersetzung der Zeichenketten von Typnamen
- Übertragung von Informationen für den Vererbungsbaum.

Die Dekodierung berücksichtigt diese Optimierungen und erzeugt umgekehrt aus dem Bitstrom ein Schema.

Die Erfindung wird nachfolgend anhand von in den Zeichnungen dargestellten Ausführungsbeispielen erläutert. Dabei zeigt

Figur 1 eine Prinzipdarstellung zu Erläuterung der erfindungsgemäßen Encodierung/Decodierung,

Figur 2 eine Darstellung zur Erläuterung der Details einer bevorzugten Ausführungsform der Erfindung,

Figur 3 eine Darstellung zur Erläuterung der Details einer weiteren bevorzugten Ausführungsform der Erfindung und

Figur 4 eine Prinzipdarstellung einer bevorzugten Ausführungsform eines erfindungsgemäßen Decoders.

Da XML Schemas ihrerseits XML Dokumente sind, denen eine standardisierte Syntaxdefinition zugrunde liegt, nämlich ein sogenanntes "Schema for Schemas" (W3C Spezifikation), das quasi ein Metaschema darstellt, kann ein Schema ebenfalls mit dem oben genannten BiM-Verfahren codiert und übertragen werden.

In Figur 1 ist eine Anordnung gezeigt, bei der, in einem ersten Schritt, mit einem BiM-Encodierverfahren BiM-E aus einem XML-Schema XMLS in Abhängigkeit eines Metaschemas SS einen Teil eines Bitstromes oder einen Bitstrom BS1 erzeugt wird und bei der, in einem zweiten Schritt, mit dem selben BiM-Encodierverfahren BiM-E aus einem XML-Dokument XML in Abhängigkeit des Schemas XMLS ein weiterer Teil des Bitstromes oder ein Bitstrom BS2 erzeugt wird sowie in umgekehrter Richtung mit einem BiM-Decodierverfahren BiM-D aus den beiden Teile des Bitstromes oder aus den Bitströmen BS1 und BS 2 ein XML Schema und das XML-Dokument wiedergewonnen werden.

In einer ersten bevorzugten Ausgestaltung der Erfindung wird eine Abspaltung von sogenannten „anonymous Types" aus der Element- bzw. Attributdeklaration vorgenommen.

Die Übertragung eines XML Dokuments erfolgt beim BiM-Verfahren "depth first", der Vorgang der Schema Kompilierung am Decoder verlangt aber einen Aufbau "breadth first", wobei

die diese Ausdrücke bspw. auf der Internetseite http://www.generation5.org/simple_search.shtml näher erläutert sind. Bei Gruppen wie Sequence oder Choice kann dies durch einen kleinen Zwischenspeicher auf Decoderseite ausgeglichen werden, aber bei den "anonymous Types", die den Typ eines einzelnen Elements oder Attributs definieren können rechtfertigt der Aufwand eine Umstrukturierung auf Encoderseite: die anonymous Type Definitionen, im nachfolgenden Beispiel mit AT0 bezeichnet, werden aus der Elementdeklaration des Elements " CurriculumVitae" herausgelöst und erhalten einen Namen und/oder Code, der zur Referenzierung bei dem entsprechenden Element verwendet wird. Vorteilhafterweise wird hierdurch die Tiefe der Hierarchie der übertragenen Typen reduziert, wodurch die Kompilierung des Schemas auf der Decoderseite vereinfacht wird.

Beispiel:

Schema vor der Umstrukturierung

```
<complexType name="PersonDescriptor">
   <element name="CurriculumVitae">
      <complexType>
         <element name="name" type="string"/>
         <element name="birthday" type="date"/>
            ...
      </complexType>
   </element>
   <element name="profession" type="profTp" />
</complexType>
```

Schema nach der Umstrukturierung

```
<complexType name="PersonDescriptor">
   <element name="CurriculumVitae" type="AT0"/>
   <element name="profession" type="profTp" />
```

```
</complexType>

<complexType name="AT0">
   <element name="name" type="string"/>
   <element name="birthday" type="date"/>
            ...
</complexType>
```

In einer zweiten bevorzugten Ausgestaltung der Erfindung wird die Normalisierung der Syntax Trees, wie sie in BiM spezifiziert ist, auf der Encoderseite durchgeführt.

Im BiM-Verfahren werden sogenannte „Finite State Automatons" die zur Dekodierung des Bitstroms verwendet werden aus Syntax Trees erzeugt, welche die Struktur des XML Schemas abbilden. Um die Codiereffizienz zu steigern entsprechen diese Syntax Trees nicht 1:1 den textuellen XML Definitionen, sondern es werden Normalisierungen vorgenommen. Drei verschiedene Fälle können hierbei auftreten:

1. Vereinfachung einer Gruppe, die nur ein Element enthält:
Die Gruppe wird aufgelöst, und das enthaltene Element wird auf der Ebene der aufgelösten Gruppe in das Content Modell einsortiert, wobei die Attribute minOccurs und maxOccurs des Elements durch das Produkt der entsprechenden Attribute der aufgelösten Gruppe und des Elements vor der Umgruppierung ersetzt wird.

2. Vereinfachung einer choice-Gruppe, die ein Element mit dem Attributwert minOccurs=0 enthält:
Das Attribut „minOccurs" der choice Gruppe wird unabhängig vom vorhergehenden Wert auf 0 gesetzt, das Element, das einen Attributwert minOccurs=0 hatte, wird einen Attributwert minOccurs=1 zugewiesen.

3. Vereinfachung von verschachtelten choice-Gruppen:
Enthält eine choice Gruppe eine andere choice Gruppe, die die Attributwerte minOccurs=maxOccurs=1 enthält, so wird diese choice Gruppe aufgelöst, und der Inhalt direkt der darüberliegenden choice Gruppe eingegliedert.

Diese Vereinfachungen sollten bei der Übertragung des Schemas schon am Encoder vorgenommen werden, da die Syntax Tree Transformationen die Vergabe der normativen Codes beeinflusst, und die Kompilierung des Schemas auf Decoderseite vereinfacht wird, wenn das Content-Modell direkt übernommen werden kann.

Die Vorteile liegen hier darin, dass hierdurch ebenfalls der Decoder entlastet wird und das Content-Modell direkt wie es bei der Typdecodierung entsteht dem Schema-Compiler zugeführt werden kann.

In einer dritten bevorzugten Ausgestaltung der Erfindung wird,
wie in Figur 2 gezeigt, eine Ersetzung der Zeichenketten von Typnamen durchgeführt.

Im Attribut "name" und "base" einer Typdefinition, sowie beim Attribut "type" einer Element- oder Attributdeklaration treten häufig im Schema die selben Typnamen auf, die als Zeichenkette mehrfach übertragen werden würden. Bei der Codierung von Typnamen ist es deshalb vorteilhaft anstatt des Namens nur eine Nummer zu codieren, und separat dazu eine Tabelle, welche die Nummern wieder zu den ürsprünglichen Namen in Beziehung setzt. Als Nummer bietet sich die Typnummer an, die der unten noch näher erläuterte Vererbungsbaum des Ur-Typs allen complexTypes zuordnet.

Entsprechendes gilt auch für das Attribut „name" von globalen Element-Deklarationen und deren Referenzen in „ref"-

Attributen und für den Namen von Ersetzungsgruppen im Attribut "substitutionGroup". In diesen Fällen kann beispielsweise der Schemaverzweigungscode SBC der globalen Elemente verwendet werden.

Hiermit kann Datenvolumen eingespart werden, da eine wiederholte Referenzierung auf den selben Typnamen kompakter dargestellt werden kann und die Typzuordnungstabelle mit einem Standardkompressor besser komprimiert werden kann, da die Typnamen nicht über den Bitstrom verteilt auftreten, sondern kompakt in einem zusammenhängenden Bereich im Bitstrom.

In einer vierten bevorzugten Ausgestaltung der Erfindung erfolgt eine Übertragung von Informationen für den Vererbungsbaum.

Jede Typdefinition enthält im sogenannten Attribut "base", falls es vorhanden ist, die Information von welchem Typ er vererbt worden ist. Wenn alle diese Informationen für ein Schema gesammelt werden, ergibt sich eine Baumstruktur, der sogenannte Vererbungsbaum. Der Vererbungsbaum wird beim BiM-Codierungsverfahren verwendet, um im Falle einer Typumandlung (type-cast) den neuen Typ des Elements zu übermitteln. Dabei ist der Code der allen vom Basistyp vererbten Typen zugeordnet wird, also der sogenannte Type Code, sowie die Länge dieses Codes für eine korrekte Dekodierung entscheidend. Die Länge ergibt sich aus der Gesamtzahl aller Typen im Vererbungsbaum unter dem Basistyp. Wenn das Schema vollständig übertragen wurde lassen sich sowohl die Codes als auch die Codelänge auf der Decoderseite eindeutig ermitteln. Wenn aber das Schema auf der Decoderseite nicht vollständig ist, muss noch Zusatzinformation übertragen werden, um bereits übertragenen Typen Type Codes zuzuweisen.

Jeder übertragene Typ hat im Namensfeld die Nummer des Typecode bezogen auf den Urtyp. Damit lässt sich der Typecode der abgeleiteten Typen durch einfache Differenzbildung ermitteln. Es fehlt noch die Information über die Mächtigkeit des durch den übertragenen Typen definierten Unterbaums, und damit die Länge der Typecodes der von diesem übertragenen Typen abgeleiteten Typen. Diese Länge lässt sich mit wenigen Bits in einem variablen Längencode übertragen.

In Figur 3 ist beispielhaft ein Vererbungsbaum eines Schemas mit dem Typ A, von dem weitere Typen abgeleited sind, dargestellt. Dieser Typ bekommt bezüglich des Urtyps "anyType" beispielsweise den Typecode 134. Von Typ A sind die Typen AA, AB und AC abgeleitet, deren Typecodes bezüglich des Urtyps angegeben sind. Um den Typecode bezüglich des Basistyps A zu ermitteln, genügt es vom Typecode des gewünschten Typs den Typecode des Basistyps und eins zu subtrahieren:

$$TC_{Type} = TC_{Type\ bzgl.\ Urtyp} - TC_{Basistyp\ bzgl.\ Urtyp} - 1$$

Die fehlende Information über die Länge des Typecodes läßt sich am besten in der Referenztabelle als zusätzliche Zahl integrieren.

Um die Information in der Typzuordnungstabelle mit einem Standardkompressor komprimieren zu können empfiehlt es sich, sie auf ganze Bytes ausgerichtet abzulegen (bytealigned). Die erste Zahl ist eine vluimsbf5 Zahl, die die Zahl der Zeilen in der Tabelle codiert, dann folgt eine vluimsbf5 Zahl, die die Nummer an Bits für den Typecode codiert, und eine weitere vluimsbf5 Zahl, die den Typecode bzgl. des Urtyps selbst darstellt. Es folgen Füllbits oder Stuffing Bits um die Ausrichtung auf Bytegrenzen zu erreichen.

Format der Typzuordnungstabelle			
Vuimsbf5	**Vuimsbf5**	**Bits**	**Zeichenkette**
Zahl der Zeilen			
Länge Typecode 1	Typecode 1	0-7 Füllbits	Name Typ 1
Länge Typecode 2	Typecode 2	0-7 Füllbits	Name Typ 2
...	...	...	...

Die Übertragung einer Typzuordnungstabelle ermöglicht es, die in einem kodierten Dokument evtl. vorhandenen Typecodes korrekt zu decodieren, auch wenn das zugrundeliegende Schema nicht oder noch nicht vollständig übertragen und/oder decodiert wurde.

Entsprechend sind mit globalen Elementen der globale SBC und bei Elementen, die zu einer Ersetzungsgruppe gehören, der Ersetzungscode zu übermitteln, wobei vorab für alle globalen Elemente einmal die globale SBC-Länge und mit dem Kopfelement der Ersetzungsgruppe die Länge des jeweiligen Ersetzungscodes übermittelt werden.

Es ist jede Kombination der in den einzelnen Ausgestaltungen dargestellten Merkmale bei der Encodierung möglich und kann in entsprechender Weise auch bei der Decodierung Eingang finden.

Das BiM-Verfahren erfordert es, dass das XML-Schema in ein Format kompiliert wird, das die Bestimmung der Länge der Codeworte und die Auswahl der Datenelemente durch die Werte der Codes gestattet. Dafür gibt es mehrere Möglichkeiten. Im MPEG-7 Standard (ISO/IEC 15938-1:2001 Part1: Systems bzw.

ISO/IEC 15938-6:2001 Part6: Referenzsoftware) ist für die Decodierung der Nutzlast bzw. Payload ein Modell vorgeschlagen, das Endliche Zustandsautomaten (Finite State Automatons) verwendet, und für die Decodierung eines Context Pfades Codetabellen, die aus dem Schema generiert werden.

In einer in Figur 4 dargestellten bevorzugten Ausgestaltung des erfindungsgemäßen Decoders wird der Decodiervorgang mit einem Bytecodemodell beschrieben, wobei die Schemastruktur in ein System aus vernetzten Zuständen übersetzt wird, die von einem Bytecodeinterpreter BCI abgearbeitet werden, wobei ein vom Encoder empfangener Bitstrom BS die Information über den auszuwählenden Folgezustand enthält. Im Unterschied zu dem Modell, das im MPEG-7 Standard vorgeschlagen wird, ist das Bytecodemodell so angelegt, dass sowohl ein Bitstrom, der eine Payload repräsentiert, als auch ein Bitstrom der einen Context Pfad darstellt decodiert werden kann. Es ist deshalb nicht erforderlich dieselbe Information, die im Schema enthalten ist zweimal für die verschiedenen Codierverfahren am Decoder vorzuhalten. Der Interpreter BCI liest die Information aus dem Eingangsbitstrom, die ein XML Dokument oder ein XML Schema im BiM Format codiert. Diese Information erlaubt die Auswahl unter den Folgezuständen des aktuellen Zustandes, der im Bytecode abgelegt ist. Die Folgezustände sind innerhalb des Bytecodes als Pointer P fest angelegt. Je nach Konfiguration wird ein Pfad, eine Payload oder ein Bytecode ausgegeben.

Die Decodierung eines Schemas läßt sich mit den oben vorgeschlagenen Modifikationen ebenfalls effizient im Bytecodemodell realisieren. In diesem Fall wird keine Payload und kein Pfad ausgegeben, sondern direkt Bytecode erzeugt, der vom Bytecodeinterpreter für die Decodierung der entsprechenden Typen verwendet werden kann.

Der Bytecode setzt sich aus Strukturelementen bzw. den Zuständen zusammen. Die Zustände sind von verschiedenem Typ,

der mit dem Headerbitfeld des Zustandes identifiziert wird. Die Zustände enthalten abhängig vom Typ verschiedene Informationsfelder, die vom Bytecodeinterpreter gelesen, und je nach Konfiguration (Payload/ Context Pfad) und aktuellem Zustand ausgewertet werden.

Für die Arten von Zuständen, welche die Schemainformation repräsentieren sind mehrere Varianten denkbar. Wesentlich ist, dass sich durch die Zustände des Bytecodemodells alle Syntaxelemente eines XML Schemas nachbilden lassen, und dass die gesamte Information, die zur effizienten Decodierung der beiden im MPEG-7 Standard definierten Algorithmen (Context Pfad/Payload) notwendig ist, in den Zuständen zur Verfügung gestellt wird.

Ein möglicher Aufbau des Bytecodes wird im folgenden kurz dargestellt.

Arten von Zuständen, Übersicht:

1. Kopfzustand eines complexTyps

Der Kopfzustand eines Typs bildet den Einsprungspunkt bei der Decodierung eines complexType. Er enthält den Namen des Typs (falls es sich nicht um einen anonymen Typ handelt) sowie Information zu Vererbung des Typs (Zeiger auf Basiszustand) sowie Polymorphismus.
Spezifisch für die Payloadcodierung ist ein Zeiger auf eine Liste der Attribute des Typs. Spezifisch für die Context Pfad Codierung sind Felder mit der Zahl der Kindelemente für die Context - und Operand Tree Branch Code Tabellen.
Das letzte Informationsfeld ist ein Zeiger auf den Folgezustand, d.h. der erste Zustand, der den Inhalt des complexTypes repräsentiert (beispielsweise ein Elementzustand oder ein Auswahlzustand).

Graphische Darstellung eines Kopfzustands:

Headerbitfeld
Pointer auf String mit Name
Pointer auf Kopfzust. Basistyp
Pointer auf Vererbungsbaum
Zahl der Kinder Context TBC
Zahl der Kinder Operand TBC
Pointer auf Folgezustand

2. Auswahlzustand

Ein Auswahlzustand bildet eine choice Gruppe des XML Schemas nach. Der Auswahlzustand enthält im wesentlichen eine Pointerliste mit möglichen Folgezuständen. Um den tatsächlich ausgewählten Zustand zu bestimmen muß bei der Decodierung einer Payload der Bitstrom gelesen werden. Vom Auswahlzustand gibt es zwei Varianten: einen Startzustand, der in die verschiedenen möglichen Folgezustände verzweigt, sowie einen Endzustand, der die Auswahl wieder zusammenfaßt.

3. Elementzustand

Der Elementzustand bildet eine Elementdeklaration in einem complexType eines Schemas nach. Er enthält einen Pointer auf eine Zeichenkette mit dem Namen des Elements, sowie einen Pointer auf den Kopfzustand des Typs. Ferner ist evtl. Information über die Länge des Position Codes (nur für Pfad-Decodierung) und für Substitution Groups vorhanden.

4. Attributzustand

Ein Attributzustand bildet eine Attributdeklaration eines Schemas nach. Enthalten sind ein Pointer auf den Namen des Attributs, sowie ein Pointer auf den Kopfzustand des simpleType des Attributs.

5. Occurrencezustand

Ein Occurrencezustand bildet die minOccurs und maxOccurs Attribute nach, die bei einem XML Schema z.B. bei einem Element oder einer Gruppe (choice, sequence, ...) auftreten können. Er enthält einen Zeiger auf den Folgezustand, falls eine weitere Instanz des Elements oder der Gruppe auftritt, sowie einen Zeiger auf den Folgezustand, falls die letzte Instanz der Gruppe codiert wurde. Da bei XML Schemas die Möglichkeit besteht, daß ein Element sich selbst enthält (in der complexType Definition des Elements, oder in einer noch tieferen Verschachtelung tritt das Element selbst wieder auf) kann auch ein Occurrencezustand gleichzeitig mehr als einmal aktiv sein. Deshalb ist ein Zeiger auf einen Stapel innerhalb des Occurrencezustands erforderlich, die den aktuellen Zustand jeder aktiven Instanz des Occurrencezustands sichert.

6. Endzustand eines Typs

Der Endzustand eines Typs enthält eine Zeigerliste mit allen Attributen dieses Typs. Sie ist bei der Decodierung eines Pfades erforderlich, da in den Tree Branch Code Tabellen alle Attribute am Ende der Tabelle einsortiert werden. Beim erreichen eines Endzustands verzweigt der Bytecodeinterpreter hierarchisch in das Element, das diesen Typ aufgerufen hat. Die entsprechende Information über das aufrufende Element muß im Arbeitsspeicher des Bytecodeinterpreters abgelegt sein.

7. Kopfzustand eines simpleTypes

Dieser Zustand steuert die Decodierung von Inhalt, d.h. er enthält einen Pointer auf einen Codec, der spezifisch Daten des betreffenden Typs aus dem Bitstrom lesen und decodieren kann. Der Typ des Codecs ist in einem Informationsfeld spezifiziert.

Die wesentlichen Vorteile des Bytecodemodells im Vergleich zum Stand der MPEG-7 Referenzsoftware sind:

1. Die Schemainformation wird für beide Codierverfahren (Context Pfad / Payload) nur einmal am Decoder repräsentiert. Der größte Teil der Information in den Bytecodezuständen sind für beide Verfahren relevant. Ein kleinerer Teil ist spezifisch für jeweils eines der beiden Verfahren. Deshalb ist die Darstellung der Schemainformation am Decoder sehr kompakt.

2. Das Bytecodemodell stellt ein wohldefiniertes Datenformat für Schemainformation zur Verfügung, das sich z.B. auch zum Vorkompilieren und Abspeichern eignet (anstatt dem XML-Schema als Text).

3. Die Ausführung des Bytecodes durch einen Standardprozessor kann sehr schnell erfolgen, da das Bytecodemodell den Decodiervorgang sehr gut vorbereitet. Alle Information ist direkt im Zustand über Zeiger verfügbar, und muß nicht (wie in ISO/IEC 15938-6, Part 6: Referenzsoftware) zum Teil erst in Listen gesucht werden.

Ein entsprechender Encoder kann auf die selbe Art und Weise realisiert werden, wobei er in der Weise invers ist, als dass die Zustände von der textuellen Repräsentation des strukturierten Dokuments gesteuert werden und die Zustandsübergänge die binäre Repräsentation generieren.

Patentansprüche

1. Verfahren zum Encodieren von strukturierten Dokumenten, insbesondere XML-Dokumenten,
bei dem, in einem ersten Schritt, die Struktur des Schemas (XMLS) normiert wird, wobei Gruppen mit Elementen und/oder Attributen vereinfacht werden,
bei dem mit einem Encodierverfahren (BiM-E) aus dem normierten Schema in Abhängigkeit eines Metaschemas (SS) ein Teil eines Bitstromes oder ein Bitstrom (BS1) erzeugt wird.

2. Verfahren nach Anspruch 1,
bei dem, in einem weiteren Schritt, mit dem selben Encodierverfahren (BiM-E) aus einem Dokument (XML) in Abhängigkeit des Schema (XMLS) ein weiterer Teil des Bitstromes oder ein weiterer Bitstrom (BS2) erzeugt wird.

3. Verfahren nach Anspruch 1 oder 2,
bei dem Elementdeklarationen und/oder Attributdeklarationen der Schemadefinition eines strukturierten Dokuments derart umstrukturiert werden, dass anonyme Typdefinitionen (AT0) aus den Elementdeklarationen und/oder Attributdeklarationen herausgelöst werden und einen Namen und/oder Code erhalten, der zur Referenzierung bei dem entsprechenden Element verwendet wird.

4. Verfahren nach einem der Ansprüche 1 bis 3,
bei dem anstatt Typnamen und/oder Elementnamen und/oder Namen von Ersetzungsgruppen nur Nummern sowie eine oder mehrere Tabellen mit einer Zuordnung zwischen Nummern und Typnamen und/oder Elementnamen und/oder Namen von Ersetzungsgruppen codiert wird.

5. Verfahren nach einem der vorhergehenden Ansprüche,
bei dem Informationen für den Vererbungsbaum von Typen, globalen Elementen und/oder Ersetzungsgruppen codiert werden,

wobei jeder Typ durch eine Information über seinen Typcode bezogen auf den Urtyp und der Länge aller Typcodes, die sich auf den beschriebenen Typen beziehen, beschrieben wird und/oder jedes globale Element durch die Länge des SBC und einen SBC und/oder jedes Element in einer Ersetzungsgruppe durch die Länge der Ersetzungscodes und einen Ersetzungscode beschrieben wird.

6. Verfahren zum Decodieren von strukturierten Dokumenten, insbesondere XML-Dokumenten,
bei dem, mit einem Decodierverfahren (BiM-D) aus einem Teil eines Bitstromes oder aus einem Bitstrom (BS1) in Abhängigkeit eines Metaschemas (SS) ein Schema (XMLS) erzeugt wird,
bei dem im Bitstrom festgestellt wird, ob die Struktur des Schemas bereits normiert wurde, wobei Gruppen mit Elementen und/oder Attributen vereinfacht wurden, und für diesen Fall keine Normierung durchgeführt wird und

7. Verfahren nach Anspruch 6,
bei dem, in einem zweiten Schritt, mit dem selben Decodierverfahren (BiM-D) aus einem weiteren Teil des Bitstromes oder einem weiteren Bitstrom (BS2) in Abhängigkeit des Schema (XMLS) ein Dokument (XML) erzeugt wird.

8. Verfahren nach Anspruch 6,
bei dem, während der Decodierung des Schemas (XMLS), mit dem selben Decodierverfahren (BiM-D) aus einem weiteren Teil des Bitstromes oder einen weiteren Bitstrom (BS2) in Abhängigkeit des bereits decodierten Teils des Schemas (XMLS) ein Dokument (XML) erzeugt wird.

9. Verfahren nach einem der Ansprüche 6 bis 8,
bei dem Elementdeklarationen und/oder Attributdeklarationen eines strukturierten Dokuments derart umstrukturiert werden, dass anonyme Typen (AT0), denen zur Übertragung ein Name und/oder ein Code zugewiesen wurde, in die jeweilige

Elementdeklaration oder Attributdeklaration eingefügt werden, von der der jeweilige anonyme Typ referenziert wird.

10. Verfahren nach einem der Ansprüche 6 bis 9, bei dem aus dem Bitstrom Typnamen und/oder Elementnamen und/oder Namen von Ersetzungsgruppen über Nummern sowie einer oder mehrer Tabellen mit einer Zuordnung zwischen Nummern und Typnamen und/oder Elementnamen und/oder Namen von Ersetzungsgruppen decodiert werden.

11. Verfahren nach einem der Ansprüche 6 bis 10, bei dem zunächst aus dem Bitstrom Informationen für einen Vererbungsbaum von Typen und/oder globalen Elementen und/oder Ersetzungsgruppen decodiert werden, wobei jeder Typ durch eine Information über seinen Typcode bezogen auf den Urtyp und der Länge aller Typcodes, die sich auf den beschriebenen Typen beziehen, beschrieben wird und/oder jedes globale Element durch die Länge des SBC und einen SBC und/oder jedes Element in einer Ersetzungsgruppe durch die Länge der Ersetzungscodes und einen Ersetzungscode beschrieben wird.

12. Vorrichtung zum Encodieren von strukturierten Dokumenten, insbesondere XML-Dokumenten, bei der eine Encodiereinheit vorhanden ist, die, in einem ersten Schritt, die Struktur des Schemas (XMLS) normieren, wobei Gruppen mit Elementen und/oder Attributen vereinfacht werden, die aus dem normierten Schema in Abhängigkeit eines Metaschemas (SS) einen Teil eines Bitstromes oder einen Bitstrom (BS1) erzeugen.

13. Vorrichtung zum Decodieren von strukturierten Dokumenten, insbesondere XML-Dokumenten, bei der eine Decodiereinheit vorhanden ist,

die, aus einem Teil eines Bitstromes oder aus einem Bitstrom (BS1) in Abhängigkeit eines Metaschemas (SS) ein Schema erzeugt,
bei der im Bitstrom festgestellt wird, ob die Struktur des Schemas (XMLS) bereits normiert wurden, wobei Gruppen mit Elementen und/oder Attributen vereinfacht wurden, und für diesen Fall keine Normierung durchgeführt wird.

14. Vorrichtung nach Anspruch 12,
bei der die Encodiereinheit einen konfigurierbaren Bytecodeinterpreter aufweist, der Informationen in einem Bytecode interpretiert und der, abhängig von der Konfigurierung, aus dem strukturierten Dokument basierend auf einem Bytecode einen Code erzeugt, der einen Pfad oder eine Nutzlast repräsentiert.

15. Vorrichtung nach Anspruch 13,
bei der die Decodiereinheit einen konfigurierbaren Bytecodeinterpreter aufweist, der durch Informationen aus dem Bitstrom konfigurierbar ist und der, abhängig von der Konfigurierung, aus dem Bitstrom basierend auf einem Bytecode einen Pfad, eine Nutzlast oder einen Bytecode erzeugt.

Zusammenfassung

Verfahren und Vorrichtungen zum Encodieren/Decodieren von strukturierten Dokumenten, insbesondere von XML-Dokumenten

Die Erfindung besteht im wesentlichen darin, mit einem Encodierverfahren aus einem Schema in Abhängigkeit eines Metaschemas einen Bitstrom oder einen Teil eines Bitstromes zu erzeugen, wobei eine oder mehrere der folgenden Optimierungen durchgeführt werden:

- Abspaltung von anonymous Types aus Elementdeklarationen und Attributdeklarationen, und Codierung als eigener Typ, dessen Typdefinition als Top-Level Element in der Schema Definition instantiiert ist,
- Normalisierung der Syntax Trees auf Encoderseite,
- Ersetzung der Zeichenketten von Typnamen
- Übertragung von Informationen für den Vererbungsbaum.

Die Dekodierung berücksichtigt diese Optimierungen und erzeugt umgekehrt aus dem Bitstrom ein Schema.

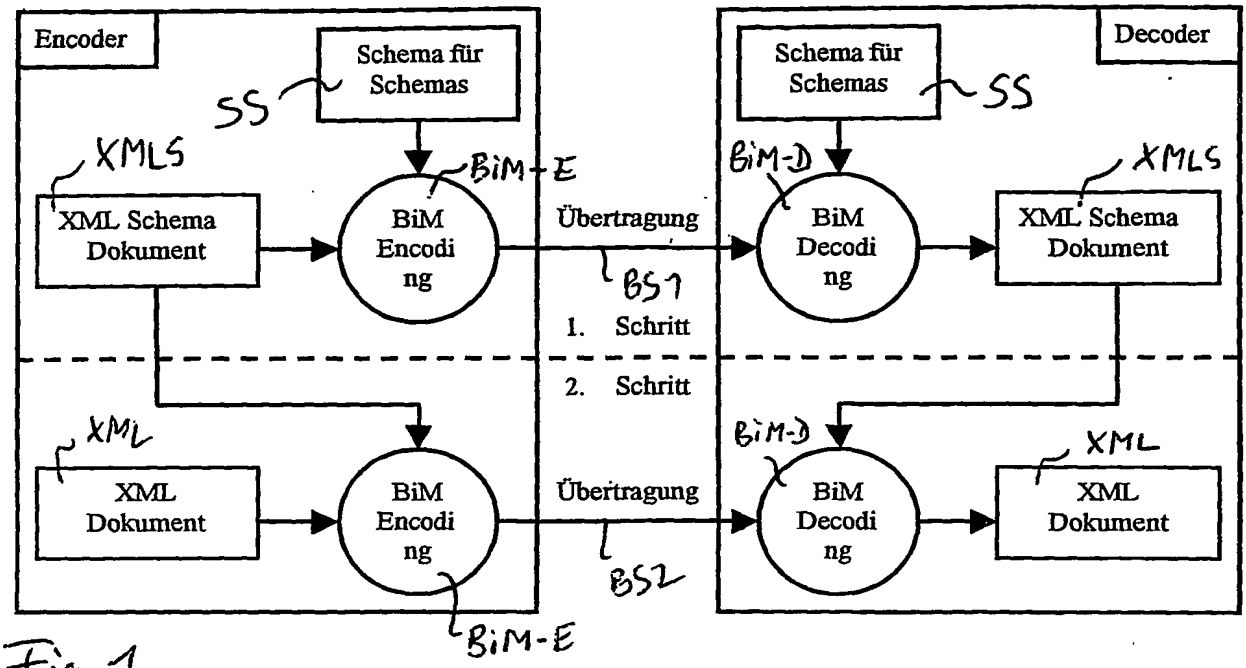

Fig. 1

```
<complexType name="42">
    <element type="15"/>
    <element type="6"/>
    <attribute type="72"/>
<complexType/>

<complexType name="13">
    <element type="42"/>
    <element type="6"/>
    <attribute type="14"/>
<complexType/>
```

Typ-nummer	Typname
42	PersonDescriptorType
15	ContactType
6	CurriculumVitaeType
72	NameType
...	...

Fig. 2

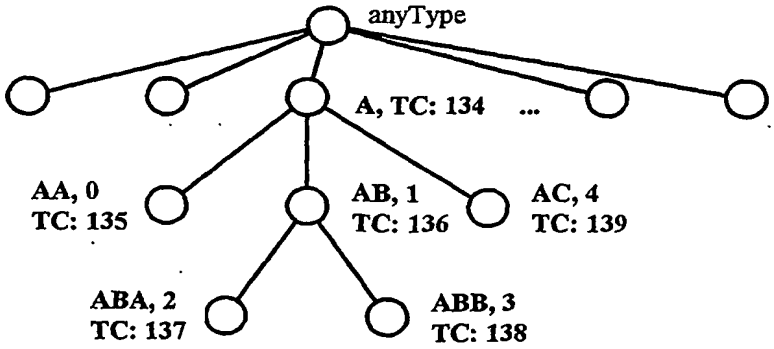

Fig. 3

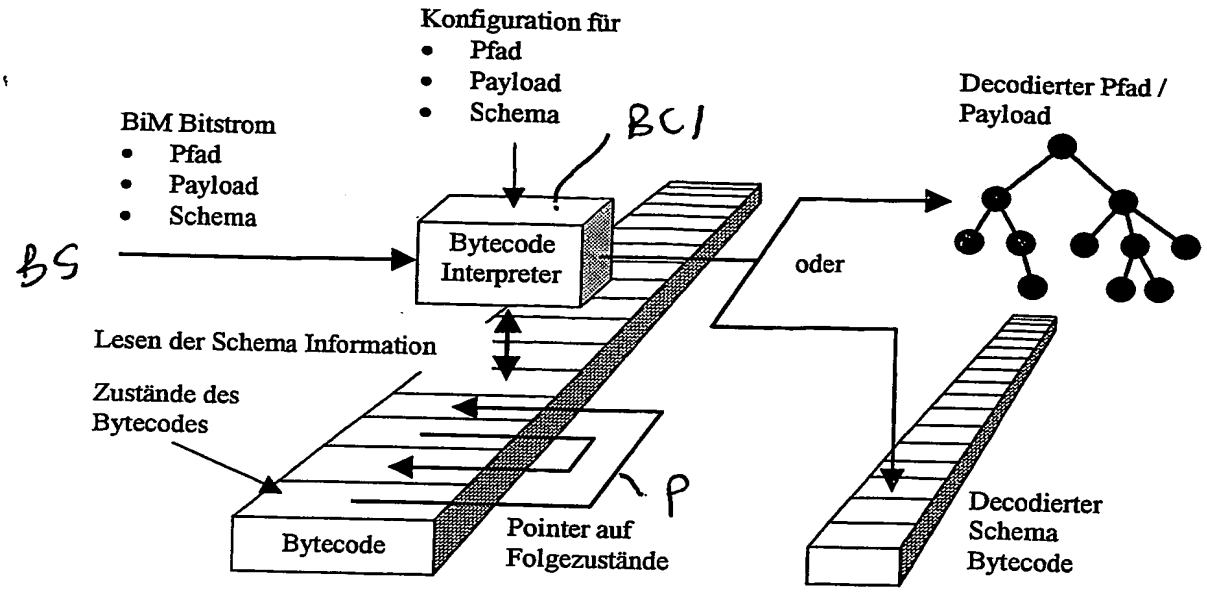
Konfiguration für
• Pfad
• Payload
• Schema
BC1
Decodierter Pfad / Payload
BiM Bitstrom
• Pfad
• Payload
• Schema
BS
Bytecode Interpreter
oder
Lesen der Schema Information
Zustände des Bytecodes
P
Bytecode
Pointer auf Folgezustände
Decodierter Schema Bytecode

Fig. 4